A Simulator for Evaluating the Leakage in Arithmetic Circuits

Audrey Lucas

A Simulator for Evaluating the Leakage in Arithmetic Circuits

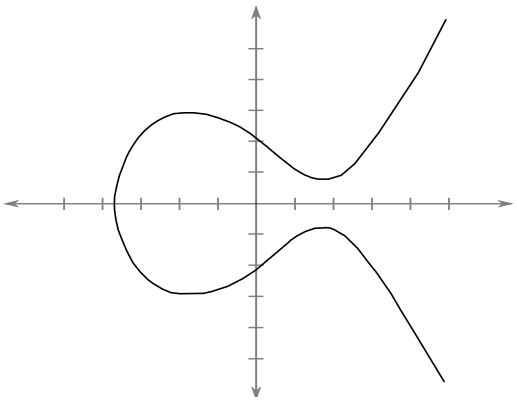
Audrey LUCAS

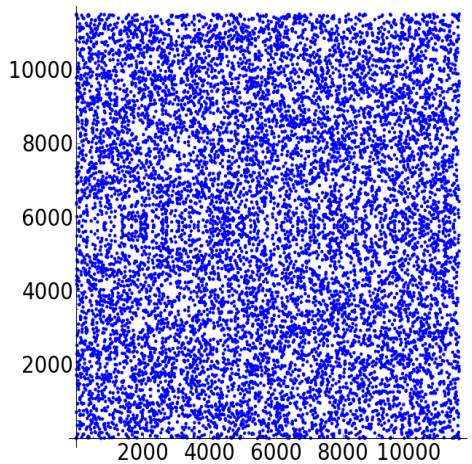
CNRS, IRISA UMR 6074

CryptArchi 2018

Outline

Elliptic Curves Cryptography (ECC) over $\mathbb{F}_p$





$$E : y^2 = x^3 + ax + b$$

Point *doubling*: DBL

$\neq$

Point *addition*: ADD

Scalar multiplication (SM)

$\Downarrow$

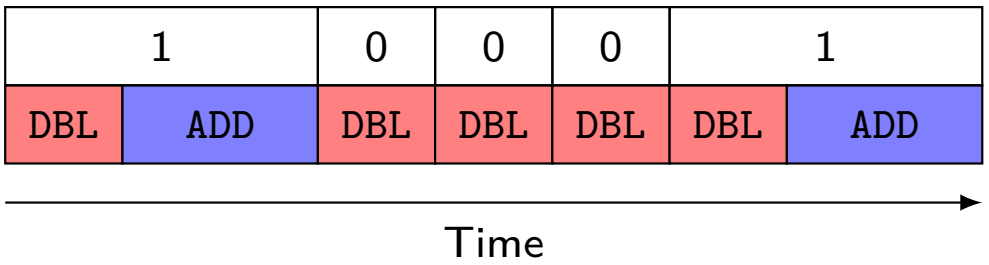
$$[k]P = \underbrace{P + P + \ldots + P}_{k \text{ times}}$$

Scalar Multiplication Example

Algorithm 1: Double and add

Input: P and $k = (k_{m-1}, \ldots, k_0)_2$
Result: $[k] \cdot P$

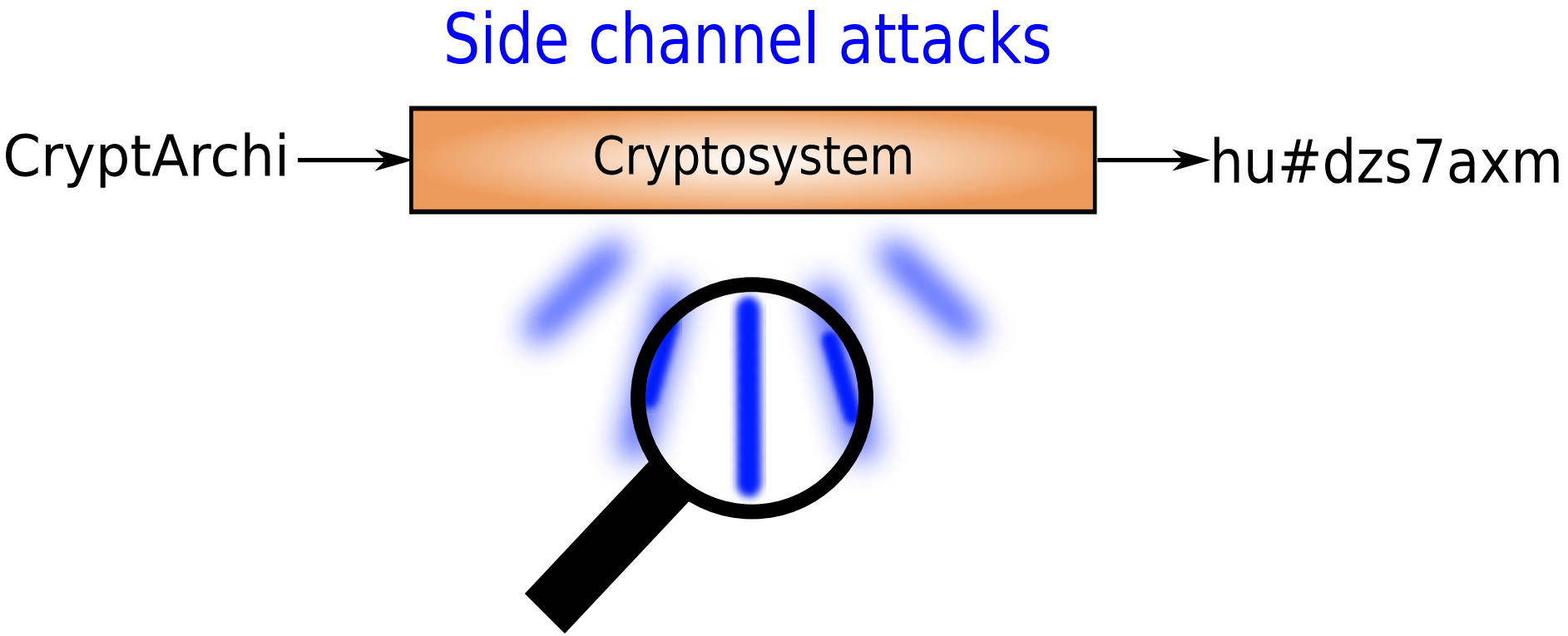
$T \leftarrow \mathcal{O}$
for $i = m - 1$ **to** 0 **do**
  $T \leftarrow 2 \cdot T$ *DBL*
  if $k_i = 1$ **then**
    $T \leftarrow T + P$ *ADD*
return T



Physical Attacks

Observation: Side Channel Attacks (SCA)

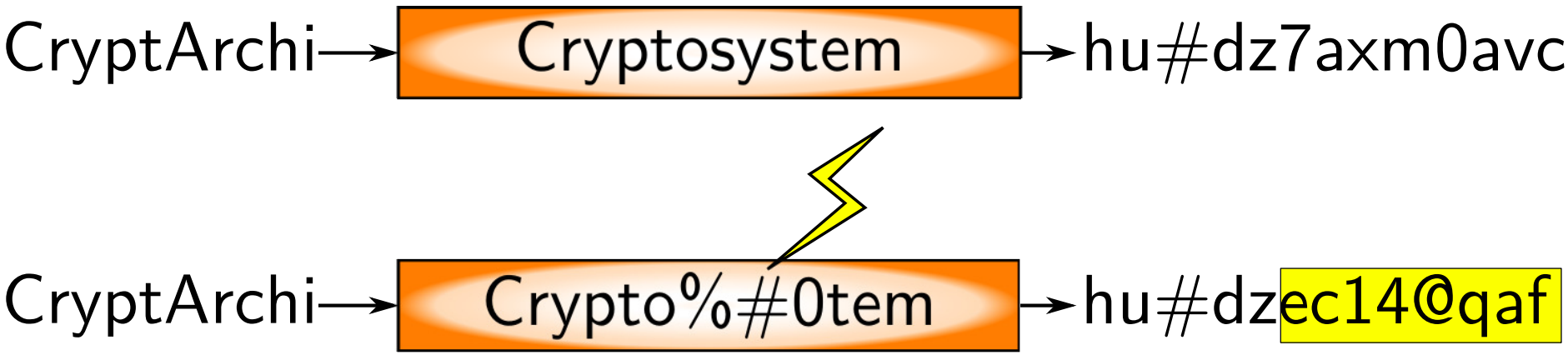
- Computation time, power consumption, electromagnetic radiation, ...
- Simple power analysis (SPA), differential power analysis (DPA), ...



Physical Attacks

Perturbation: Fault Attacks (FA)

- Clock, supply voltage, laser, ...
- Bit flip fault, stuck-at fault, ...
- Safe error, differential fault analysis (DFA), ...



Physical Attacks

Countermeasures against SCAs

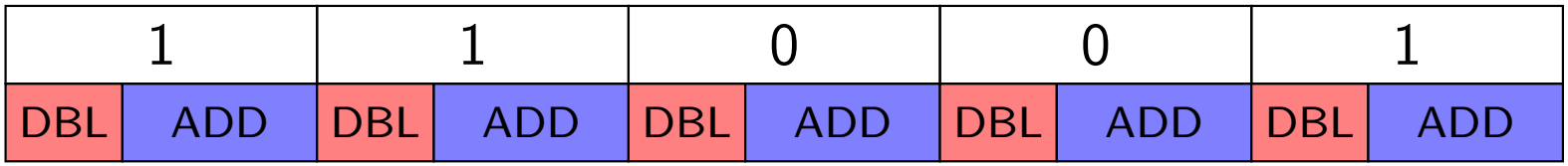
- Randomization: scalar masking, point blinding, scalar recoding, ...
- Uniformization: uniform curve, regular algorithm, ...
- Hardware: specific logic styles, reconfiguration, ...

Countermeasures against FAs

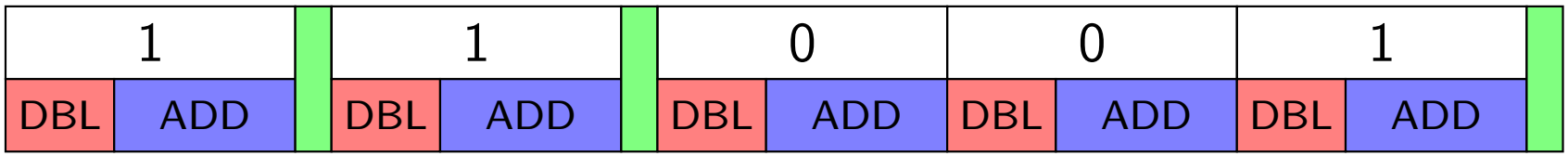
- Hardware: shielding, sensor, ...
- Redundancy calculation: time, space, ...
- ECC case: verification of point coordinates onto the elliptic curve.

Problem

Protection for one type of attacks may leave the system vulnerable on other type of attacks.



Regular SM



Regular SM and FA countermeasure

Are FA countermeasures resistant against SCA?

Outline

Simulator for Evaluating the Leakage in Arithmetic Circuits

Objective

Detection of strength/weakness of:

- Data representation (field element, point of curve)
- Computation algorithms (field and curve levels)

Attacks:

- Identify potential arithmetic leaks
- Use these leaks for preparing some SCAs (e.g. template attacks)

  attacker knows where to search in real traces

Protections:

- Help designer to locate the leaks at design time
- Countermeasures evaluation (e.g. against FA)

Preliminary

The simulator should be accurate but fast (VHDL simulations are too slow).

Typical Targeted Architecture

`w`-bit microcontroller:

- arithmetic units: adder (`wadd`), multiplier (`wmul`)
- control
- register file
- ...

Simulated Architecture for Experiment

- Focus on `w`= 32 and arithmetic units
- Target small core (1 `wadd`, 1 `wmul`)
- Can be extend to larger cores (n_a `wadd`, n_m `wmul`)

Preliminary

Implemented in Python and SageMath

Arithmetic

- Field operation modulo p (p generic)
- Montgomery representation ($\beta = 2^{32}$)
- Multiplication with Karatsuba
- Montgomery reduction

Notations

- Field addition: `fadd`
- Field multiplication: `fmul`

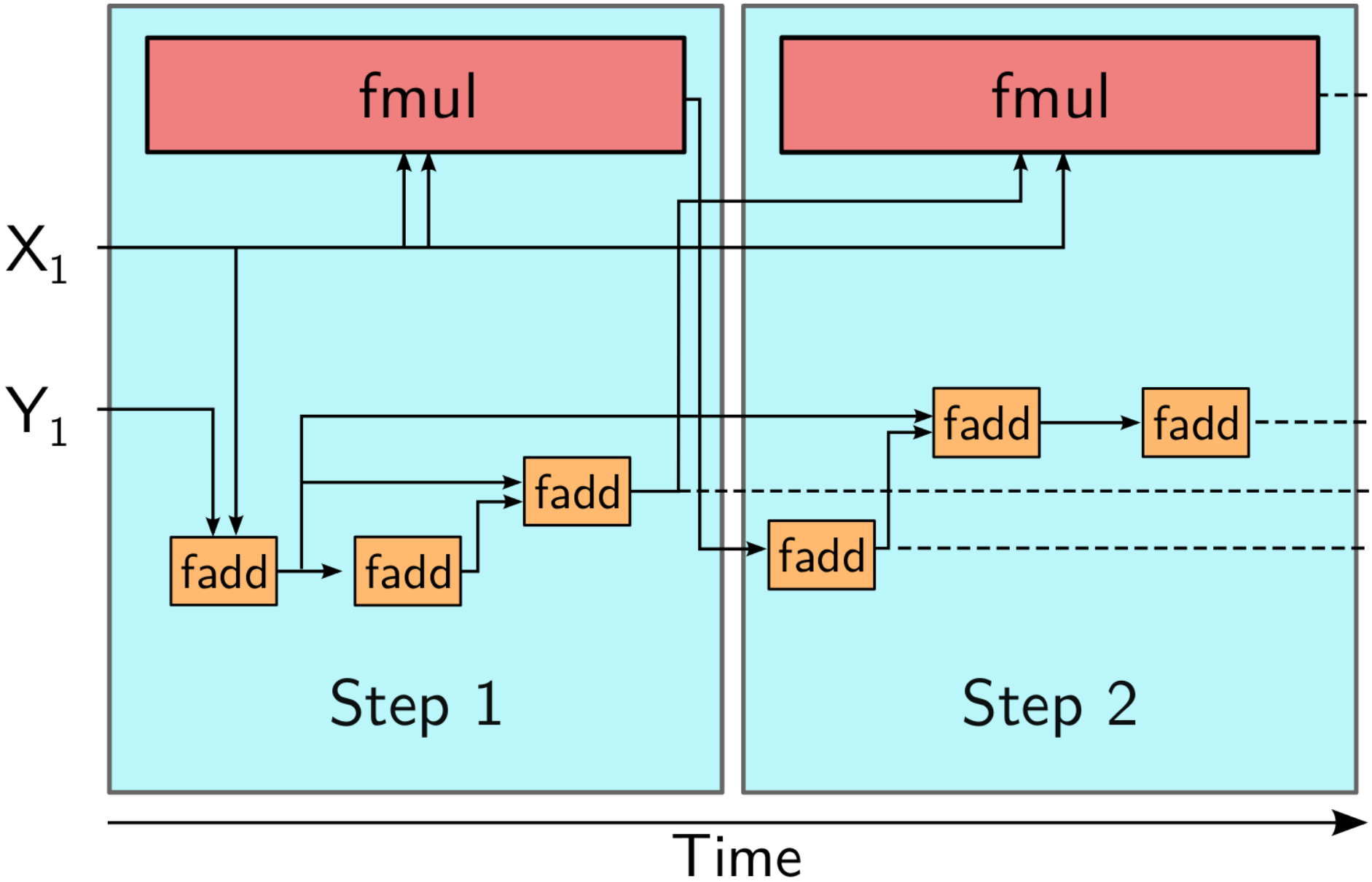
Formulas Integration

Formulas Integration

- Create a table for formulas
  - 1 line corresponds to 1 field operation
  - Field operation: 2 inputs and 1 output
- Operation scheduling according to dependencies
- Add "step" notion (latency)
  - 1 `fmul` by step
  - many `fadd` by step
- Writing code file for SM

Output	Inputs	Ope	Step
xx	X_1, X_1	`fmul`	1
T_0	X_1, Y_1	`fadd`	
T_1	T_0, T_0	`fadd`	
M	T_1, T_0	`fadd`	
A	M, M	`fmul`	2
T_2	xx, T_1	`fadd`	
B	T_2, T_0	`fadd`	
X	B, B	`fadd`	
	$\vdots$		

Formulas Integration

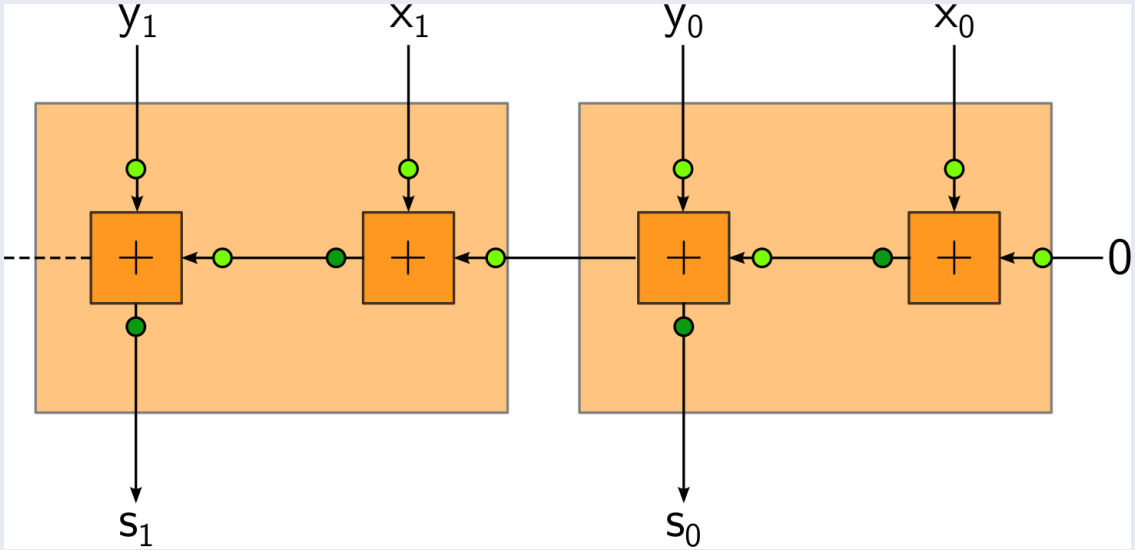


Activity Monitoring

- Each field operation uses several arithmetic units
- Recording of input and output in all arithmetic units
- Obtained activity traces for field operations

  estimated by Hamming weight (HW) variation

Field Addition Example



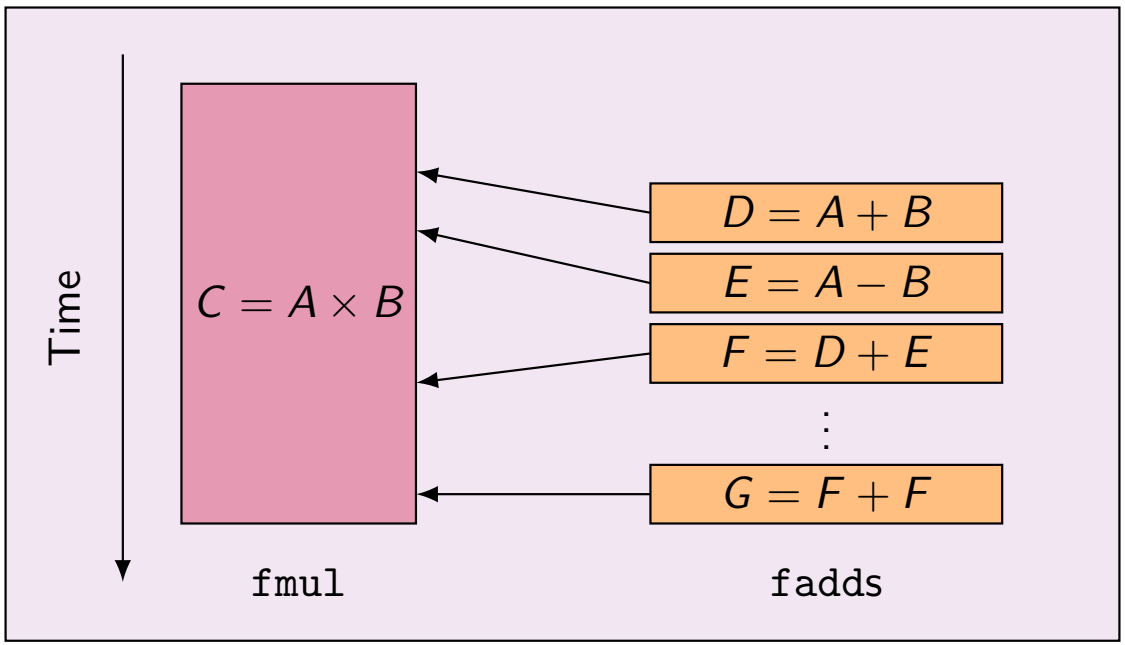
- $X = (x_0, x_1, \ldots)_{2^{32}}$
- $Y = (y_0, y_1, \ldots)_{2^{32}}$
- 32-bit words

Fusion of Traces

The global trace is constructed by fusion of field operations traces

- During `fmul`, adder is sometimes idle
- When adder is idle in `fmul`$\Rightarrow$ `fadd` is performed in parallel of `fmul`

Parallelization aspect in order to be close to processor

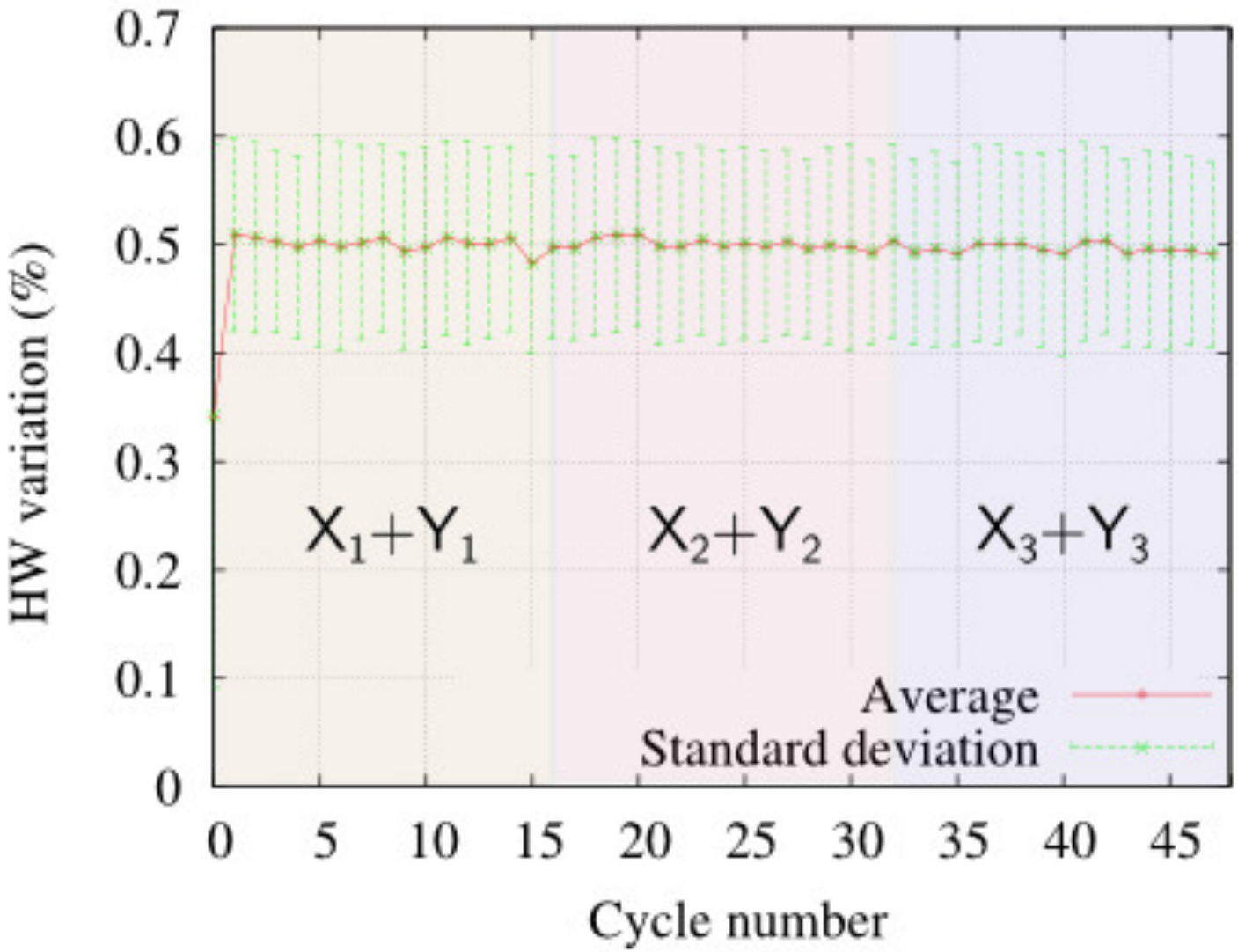


Outline

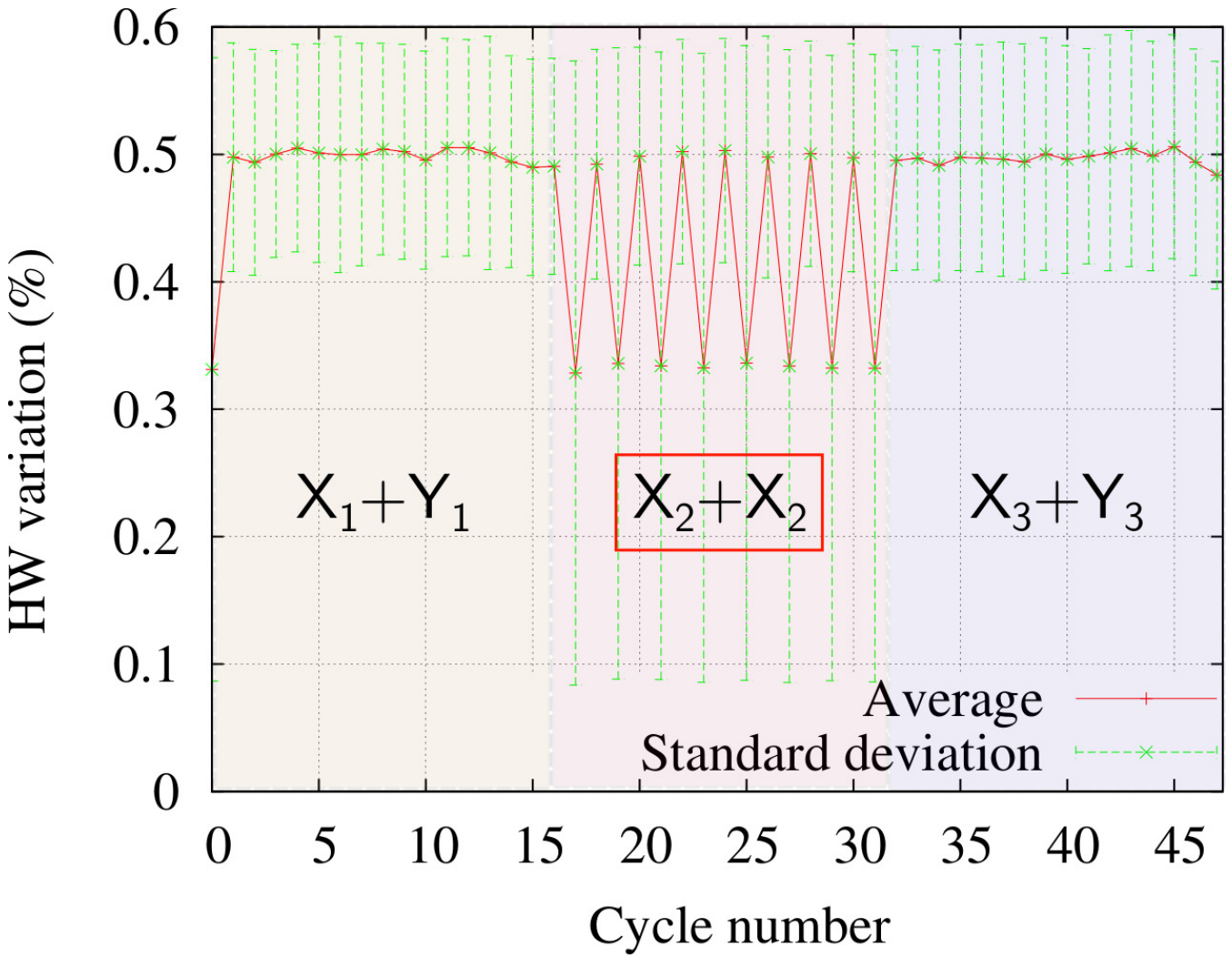
Experimentation

- Operation sequence: 3 `fadds`
- `fadd` algorithm: *µNaCl* library
  - Cryptography library for microcontrollers
  - ECC: Montgomery curve
- Random inputs

Trace of 3 Field Additions



Trace of 3 Field Additions



Discussion

Mathematical validation

Comparison between result simulation of computation with SageMath

Strengths

- Faster simulation than using VHDL description
  - data width $>$ 100-bit $\Rightarrow$ Very slow in VHDL
- Simulator can be configurable
- Adaptable to many curves, algorithms and mathematical objects representations
- Adaptable to various numbers of `wadd` and `wmul`

Future work

Calibration of the architecture model with real measurement

Outline

Conclusion

What is done

Low level arithmetic simulator:

- for Weierstrass curve
- for Montgomery curve

Future works

- Architecture model calibration
- Implementation and evaluation of protections against FA
- Use the simulator for prepare and optimize attacks

Thank you for your attention.

Questions?